




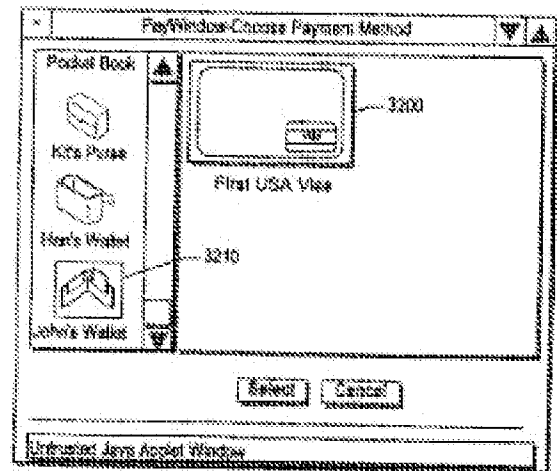


# 1. A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR NETWORK ELECTRONIC AUTHORIZATION UTILIZING AN AUTHORIZATION INSTRUMENT

Bibliographic data	Description	Claims	Abstract	Original document	INPADOC legal status
<b>Publication number:</b>	JP2000512405 (T)				<b>Also published as:</b>
<b>Publication date:</b>	2000-09-19				 WO9741540 (A1)
<b>Inventor(s):</b>					 JP2008186474 (A)
<b>Applicant(s):</b>					 EP0901672 (A1)
<b>Classification:</b>					 EP0901672 (B1)
<b>- international:</b>	G06Q30/00; G06Q10/00; G06Q20/00; G06Q50/00; G06Q30/00; G06Q10/00; G06Q20/00; G06Q50/00; (IPC1-7): G06F17/60; G06F19/00				 DE69726138 (T2)
<b>- European:</b>	G06Q20/00K2B; G06Q20/00K3B; G06Q20/00K4C				
<b>Application number:</b>	JP19970539051T 19970424				more >>
<b>Priority number(s):</b>	US19960638355 19960426; US19960641992 19960426; WO1997US06938 19970424				
<a href="#">View INPADOC patent family</a>					
<a href="#">View list of citing documents</a>					
<a href="#">Report a data error here</a>					
Abstract not available for JP 2000512405 (T)					
Abstract of corresponding document: <b>WO 9741540 (A1)</b>					
<a href="#">Translate this text</a>					

An electronic monetary system provides for transactions utilizing an electronic monetary system that emulates a wallet or a purse that is customarily used for keeping money, credit cards and other forms of payment organized. Access to the instruments in the wallet or purse is restricted by a password to avoid unauthorized payments. When access is authorized, a graphical representation of the payment instruments is presented on the display to enable a user to select a payment method of their choice. Once a payment instrument is selected, a summary of the goods for purchase is presented to the user and the user enters their electronic approval for the transaction or cancels the transaction. Electronic approval results in the generation of an electronic transaction to complete the order.



.....  
Data supplied from the **espacenet** database — Worldwide

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表2000-512405

(P2000-512405A)

(43) 公表日 平成12年9月19日 (2000.9.19)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テ-マコード (参考)
G 0 6 F 17/60		G 0 6 F 15/21	3 3 0
19/00		15/30	L
		15/21	3 4 0 A

審査請求 未請求 予備審査請求 有 (全 97 頁)

(21) 出願番号 特願平9-539051  
 (86) (22) 出願日 平成9年4月24日 (1997.4.24)  
 (85) 翻訳文提出日 平成10年10月26日 (1998.10.26)  
 (86) 国際出願番号 PCT/US97/06938  
 (87) 国際公開番号 WO97/41640  
 (87) 国際公開日 平成9年11月6日 (1997.11.6)  
 (31) 優先権主張番号 08/638, 355  
 (32) 優先日 平成8年4月26日 (1996.4.26)  
 (33) 優先権主張国 米国 (US)  
 (31) 優先権主張番号 08/641, 992  
 (32) 優先日 平成8年4月26日 (1996.4.26)  
 (33) 優先権主張国 米国 (US)

(71) 出願人 ヴェリフォウン、インク  
 アメリカ合衆国カリフォルニア州95054、  
 サンタ・クララ、グレイト・アメリカ・パ  
 ークウェイ 4988番  
 (72) 発明者 ウィリアムズ、ハムフリ  
 アメリカ合衆国カリフォルニア州94306、  
 パロ・アルト、サン・ジュディーン 857  
 番  
 (72) 発明者 ヒューズ、ケヴィン  
 アメリカ合衆国カリフォルニア州94402、  
 サン・マテオ、リオンリッジ・レイン 33  
 番  
 (74) 代理人 弁理士 真田 雄造 (外2名)

最終頁に続く

(54) 【発明の名称】 認可装置を使って電子ネットワーク認可をするシステム、方法及びそれを行う機器

## (57) 【要約】

電子マネーシステムは、金、クレジットカード、及び他の組織された支払い形態を管理するために通例使用されるウォレット又は財布をエミュレートする電子マネーシステムを利用する取引のために備えられる。ウォレット又は財布内の手段へのアクセスは、認可されない支払いを避けるためにパスワードによって制限されている。アクセスが認可されるとき、支払い装置のグラフィック表示は、ユーザに支払方法を選択させるためにディスプレイ上に提示される。いったん支払い装置が選択されると、購買商品のサマリーがユーザに提示され、かつユーザは、取引の電子承認を入力し、或いはその取引をキャンセルする。電子承認すると、この注文を完了する電子取引が発生する。

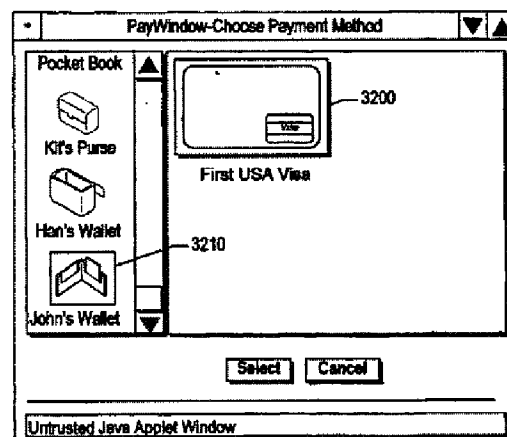


FIG.-32

## 【特許請求の範囲】

1. ネットワーク情報（110, 430）を受信しかつ送信するためのネットワークに接続された付属ディスプレイを持つコンピュータ（138, 400）において認可を開始するための方法において、
  - (a) 少なくとも1つの手段（1720, 1730）に該手段（1720）のビットマップイメージ（3200）として表示するステップと、
  - (b) 1つの手段（3200, 1720）を選択するステップと、
  - (c) 処理のために前記手段（3200, 1720）と関連した情報（1700-1792）を利用するステップと、から成る前記認可を開始するための方法。
2. この手段が、発行者選択に従い代表される請求項1に記載の方法。
3. 前記手段との取引を完了するために提案された取引及び協定のサマリーを提示する認可ディスプレイを提供するステップを含む請求項1に記載の方法。
4. 別の手段に変更し、取引を完了するための情報を得、或いはこの手段と関連した情報を調査するのを容易にするリンクを含む請求項3に記載の方法。
5. 受け取った認可を表す証印を表示するステップを含む請求項3に記載の方法。
6. ディスプレイ上で商人、発行者、広告者又は代理店にリンクを埋め込むステップを含む請求項1に記載の方法。
7. プロバイダーのプラクティスに基づいた選択のために可能にされる手段を変更する請求項1に記載の方法。
8. 一つの手段がディスプレイ上で使用された各取引のリストを表示するステップを含む請求項1に記載の方法。
9. ディスプレイ上で認可された手段に基づいたオプションのメニューを表示するステップを含む請求項1に記載の方法。
10. 選択された手段に基づいた処理を変更するステップを含む請求項1に記載の方法。
11. ネットワーク情報（134-135）を受信しかつ送信するためのネット

ワークに接続された入力装置（１２４）及び付属ディスプレイ（１３６－１３８）を持ち、ソフトウェア（１８０－１８６，３００－３８０，４００－４８０）の制御の下で、コンピュータ（１００－１３８）において、認可を開始するための装置において、

（a）少なくとも１つの手段にディスプレイ上でこの手段（７００－７９６，１０３０，１０４０及び１０２０）のビットマップイメージとして表示するソフトウェアの制御の下にあるコンピュータと、

（b）１つの手段（１０３０）を選択するソフトウェア（７００－７９６）の制御の下にあるコンピュータ（１１０）の制御の下にある入力装置（１２４）と、

（c）認可処理のために選択された手段と関連した情報を利用するソフトウェアの制御の下にあるコンピュータと、  
から成る前記認可を開始するための装置。

１２．前記手段が、発行者選択に従い代表される請求項１１に記載の装置。

１３．ソフトウェアの制御の下にある前記コンピュータが、認可されるプロセスと関連した追加の機能を提供する他のディスプレイにリンクを提供する請求項１１に記載の装置。

１４．ソフトウェアの制御の下にある前記コンピュータが、前記手段との取引を完了するために提案された取引及び協定のサマリーを提示する請求項１１に記載の装置。

１５．別の手段に変更し、取引を完了するための情報を得、或いは、この手段と関連した情報を調査することを容易にするリンクを含む請求項１１に記載の装置。

１６．ソフトウェアの制御の下にある前記コンピュータが、ディスプレイ上で１つの手段を証明するための認可コードの入力を促す請求項１４に記載の装置。

１７．ソフトウェアの制御の下にある前記コンピュータが、ディスプレイ上で受け取った認可を表す証印を表示する請求項１４に記載の装置。

１８．ソフトウェアの制御の下にある前記コンピュータが、ディスプレイ上で承

認、発行者、広告者、又は代理店へのリンクを埋め込む請求項11に記載の装置。

19. ソフトウェアの制御の下にある前記コンピュータは、ディスプレイ上の認可レベルを超える手段を表す証印を表示する請求項11に記載の装置。

20. ソフトウェアの制御の下にある前記コンピュータは、ディスプレイ上の各商人のためのデフォルト手段を限定する請求項11に記載の装置。

21. ソフトウェアの制御の下にある前記コンピュータは、一つの手段が使用された各取引のリストを表示する請求項11に記載の装置。

22. 該手段は、ディスプレイ上で発行者選択に従い代表される請求項11に記載の装置。

23. 支払い処理が、選択された手段に基づいている請求項11に記載の方法。

**【発明の詳細な説明】**

発明の名称      認可装置を使って電子ネットワーク認可をする  
システム、方法及びそれを行う機器

**発明の分野**

本発明は、通信ネットワークを通して購入する商品やサービスにアクセスするための電子支払いあるいは他の認可 (a u t h o r i z a t i o n) に関するもので、特に、クレジットカード、現金あるいは商品へのその他の支払いについて実生活での取引によく似た人間工学的な図のあるユーザネットワークインターフェースに関する。

**発明の背景**

本発明は、現金、小切手、クレジットカードや信用状に代わる経済上の交換手段として、電子通貨支払いや電子金銭送金を行うために、通貨システムを電子的に図に示すことに関する。電子通貨システムは、現金や、小切手や、カード支払いシステムと電子金銭送金との組み合わせであって、このシステムには多くの利点があるが、限界の少ないものである。このシステムは、通貨システムの加入者が経済上の価値として広く受け入れ交換するように設計された、通貨の電子的な表示を用いる。

今日、毎年、個人や機関の間で約3500億回の硬貨及び通貨取引がある。硬貨や通貨取引が広く行われていることが、購入や料金や銀行口座への入金や引き出しのような個々の取引を自動化する上で制限となっている。個々の現金取引では、正確な金額の現金を用意するあるいはそのお釣りを支払う必要がある。さらに、紙幣や硬貨を取り扱い管理することは不便であり、個人にとっても金融機関にとっても費用のかさみ、時間のかかることである。

小切手は口座の金額以内ではいかなる金額をも書き込むことができるが、小切手は流通性に限界があり、ある物理的な入れ物から出す必要がある。紙を元にした小切手システムは現金取引の限界を完全になくすものではなく、現金を扱う際に生じる多くの不便さを兼ね備えていて、小切手を処理するために不可避免的に生

じる遅れを有するものである。このために電子取引はより低いコストでより便利

になるように努めるとともに、より安全なものを目指してきた。

コンピュータを用いた電子金銭送金システム（EFT）による自動化を行うことで、大きな取引についてこれらの点のあるものは改善されてきた。電子金銭送金は、基本的にはバンキングシステムでの中央に結ばれたコンピュータ取引を通じて行う価値交換プロセスである。EFTサービスは電子「小切手」を用いる支払い送金であり、それはまず大きな商業組織で用いられている。

自動交換所（ACH）、そこではユーザが予め認可されたコードを入力して後で生じる支払い情報を入手することができる、やポイントオブセールシステム（POS）、そこでは中央コンピュータとつないで取引が行われて、その取引が即座に承認あるいは非承認される、それらは小売りや商業組織で使われているEFTシステムの例である。しかし、この種のEFTシステムを通して行う支払いは、バンキングシステムなしでは行うことができないという限界がある。さらに、ACH取引は通常営業時間外には行うことができない。

ホームバンキングを使った請求支払いサービスはEFTシステムの例であって、それを使って家庭のコンピュータから個人が支払いをすることのできるものである。現在、ホームバンキングを始めている人は非常に少ない。パーソナルコンピュータで電話線を使って支払いや口座送金や情報サービスをしている銀行はあるが、その銀行の顧客のうち1%以下の人がそのサービスを利用しているだけである。ホームバンキングがよく利用されていない一つの理由は、この種のシステムでは、顧客が必要とするお金を入金したり引き出したりすることができないことである。

商人の口座と顧客の口座間のような、口座間で金銭を送金するオンラインシステムとして用いられている、現行のEFTシステム、クレジットカードや信用状は、人間工学的インターフェースを持った自動取引システムの必要性を満たすことができない。非人間工学的インターフェースを持ったものではあるが、EFTシステムの例は米国特許第5,476,259号、第5,459,304号、第5,452,352号、第5,448,045号、第5,478,993号、第5,455,407号、第5,453,601号、5,465,291号、及び



第5, 485, 510号に示されている。

ある形をした経済上の価値を配ることのできる自動化した便利な取引を行うためには、オフライン支払いをする傾向があった。例えば、従来の現金や小切手での支払いシステムに代えるものとして、キャッシュレス支払い取引で使うことのできるある形をした「電子マネー」について、多くのアイデアが提案されている。米国特許第4, 977, 595号「電子現金を使う方法と装置」及び米国特許第4, 305, 059号「調節金銭送金システム」を参照。

もっとよく知られている技術は、所定の金額で購入した磁気ストライプカードであり、そこから特定の購入について支払った価値を差し引くことのできるものである。経済上の価値がなくなったとき、そのカードは捨てられる。他の例は、メモリーカードあるいはいわゆるスマートカードである。それは特定の購入について同様に差し引くことのできる価値を示している情報を繰り返して格納することのできるものである。

現金に代わる取引に関する技術はよく開発されているが、そのような取引について更にユーザフレンドリーなシステムや方法が強く要望されている。このようにして、ユーザから自然に見え感じられる表示を取り扱うことのできるシステムが必要とされている。特に重要なことは、一般に受け入れられている標準的な支払いと同じであることによって受け入れられるものである。

#### 実施態様の概要

発明のこれら及び他の目的について、本発明の簡単な概要を述べる。以下の概要ではいくつかの簡略化や省略を行っていることがあるが、それらは本発明の特徴を目立つようにして示すために行ったものであり、その範囲を限定するものではない。当業者が発明の概念を実施し使用することができる程度に好ましい実施態様の詳細な記述は後のところで行う。

発明の広い概念によれば、電子通貨システムは好ましい実施態様による取引を提供する。その電子通貨システムは、現金、クレジットカード、その他の支払い手段を入れておくのに伝統的に用いられている札入れ、小銭入れ、スマートカード、手帳、小切手帳、小型鞆、その他の支払い手段を入れているホルダーに代わり得るものである。札入れへのアクセスはパスワードで制限されている。パスワ

ードは暗号化することができ、あるいは機密情報へのアクセスを制限するために暗号化のキーを生み出したり、認可されていない支払いを避けるために使うことができる。アクセスが認可されたとき、ユーザデフォルト、その手段のユーザデフォルトあるいは支払い手段を入れておくホルダーのデフォルトとして、ディスプレイ上にその手段の表示（ビットマップ）がなされて、ユーザが特定の取引に使用する支払い手段を選択できるようになる。支払い手段が選択されると、購入する商品の合計が示されるので、ユーザはその取引に電子承認を与えるか取りやめるかを入力する。電子承認を与えたら、電子取引が開始されて、その注文が実行される。

#### 図面の簡単な説明

上述及び他の目的、特徴及び利点は、図面を参照しながら発明の好ましい実施態様についての以下の詳細な説明から更によく理解できるであろう。

図においては、

第1A図は、好ましい実施態様による、代表的なハードウェア環境のブロックダイアグラムである。

第1B図は、好ましい実施態様による、システムのブロックダイアグラムである。

第2図は、発明をJ A V A ベースで実施する場合に行うべき他の好ましい実施態様を示す。

第3図は、本発明の好ましい実施態様による、アーキテクチャブロックダイアグラムである。

第4図は、好ましい実施態様による、支払いマネジャーに関する詳細なロジックを示す。

第5図は、発明の好ましい実施態様で、消費者支払いメッセージシーケンスダイアグラムである。

第6図は、発明の好ましい実施態様による、支払いウィンドウユーザインターフェースの詳細なロジックを述べているフローチャートである。

第7図は、好ましい実施態様による、支払いウィンドウアドミニストレーションモードに関する詳細なロジックのフローチャートである。

第8図は、好ましい実施態様による、レジスターで処理を行うアドミニストレーションモードとレポート処理のフローチャートである。

第9図は、好ましい実施態様による、支払いウィンドウディスプレイの図である。

第10図は、好ましい実施態様による、ユーザが選択することのできる支払方法のリストの図である。

第11図は、好ましい実施態様による、認可を与えるディスプレイ画面である。

第12図は、好ましい実施態様による、支払い済み(PAID)のオーサライズを与えるディスプレイである。

第13図は、好ましい実施態様による、認可の取り消しディスプレイ画面である。

第14図は、発明の好ましい実施態様による、主画面を示す。

第15図は、好ましい実施態様による、アドミニストレーションディスプレイ画面の図である。

第16図は、好ましい実施態様による、札入れアドミニストレーションページのディスプレイを示す。

第17図は、好ましい実施態様による、支払いアドミニストレーションウィンドウである。

第18図は、好ましい実施態様による、アドミニストレーションプリファレンスページの図である。

第19図は、好ましい実施態様による、ウィンドウコントロールを示す。

第20図は、発明の好ましい実施態様による、レジスターディスプレイである。

第21図は、好ましい実施態様による、レジスターディテイルディスプレイである。

第22図は、好ましい実施態様による、レジスターファインドディスプレイである。

第23図は、好ましい実施態様による、レジスターカスタマイズディスプレイ

である。

第24図は、レジスター情報の出力対象を示すレジスター出力ディスプレイである。

第25図は、好ましい実施態様による、札入れのレポートである。

第26図は、好ましい実施態様による、レポートカスタマイズディスプレイを示す。

第27図は、好ましい実施態様による、レポート出力ディスプレイである。

第28図は、好ましい実施態様による、支払い手段の「開」「閉」アイコンを示している一組のイメージを示す。

第29図は、好ましい実施態様による、認証発行形式の図である。

第30図は、好ましい実施態様による、認証発行応答を示す。

第31図は、好ましい実施態様による、支払い手段ホルダーの一つの集合を示す。

第32図は、好ましい実施態様による、デフォルト支払い手段ビットマップを示す。

第33図は、好ましい実施態様による、カードホルダーのブランクを埋めた選択された支払い手段を示す。

そして、第34図は、発明の好ましい実施態様による、新しく定義したVISAカードを用いてコーヒを購入することを示している。

#### 詳細な説明

本発明によるシステムの好ましい実施態様は、IBM PS/2、アップルマッキントッシュコンピュータあるいはユニックスを基礎としているワークステーションのようなパーソナルコンピュータを使って行うのがよい。代表的なハードウェア環境は第1A図に示されている。それは、マイクロ処理装置のような中央処理ユニット110とシステムバス112を介して結ばれている他の多くのユニットを有する、好ましい実施態様による、ワークステーションの典型的なハードウェア構成を示している。第1図に示しているワークステーションは、ランダムアクセスメモリ(RAM)114、リードオンリーメモリ(ROM)116、ディスク格納ユニット120のような周辺機器をバス112につないでいるI/O

アダプター 118、キーボード 124 やマウス 126 やスピーカ 128 やマイクロフォン 132 及び／またはタッチスクリーン（図示せず）のような他のユーザインターフェース機器をバス 112 につないでいるユーザインターフェースアダプター 122、ワークステーションを通信ネットワーク（例えば、データ処理ネットワーク）につないでいる通信アダプター 134、及びバス 112 をディスプレイ機器につないでいるディスプレイアダプター 136 を含む。ワークステーションは通常、マイクロソフトウィンドウズオペレーティングシステム（OS）や IBM OS/2 オペレーティングシステムやマック OS またはユニックスオペレーティングシステムのようなオペレーティングシステムの上で動く。本発明はここに述べた以外のプラットフォームやオペレーティングシステムの上でも実施することができることは、当業者には明らかであろう。

好ましい実施態様は J A V A や C や C ++ 言語を用いて書かれていて、オブジェクトオリエンティッドなプログラミング手法を用いている。オブジェクトオリエンティッドなプログラミング（O O P）は複雑な用途に用いるのに広く使用されるようになってきた。O O P がソフトウェア設計や応用の主流になりつつあるので、種々なソフトウェア解法が O O P の利点を使えるように適用する必要があるだろう。メッセージングインターフェースの一連の O O P クラスとオブジェクトを与えるような電子メッセージングシステムのメッセージングインターフェースに O O P のこれらの原理を適用する必要がある。

O O P はオブジェクトを用いてコンピュータソフトウェアを開発する手段であり、それは問題を解析し、システムを設計し、プログラムを作る段階を持つ。オブジェクトは、データと、それに関係する構造と手続きの集合体との両方を含むソフトウェアパッケージである。それはデータと、構造と手続きの集合体との両方を含んでいるので、特定の仕事をを行うのに他の余分な構造、手続きあるいはデータを必要としない、自己完結成分として示すことができる。それ故に、O O P はコンピュータプログラムを、各々が特定の仕事をを行うことになっているほとんど自律的な成分、いわゆるオブジェクトの集合体とする。データ、構造及び手続きをまとめて一つの成分あるいはモジュールにパッケージングするこの概念はカプセル化と呼ぶ。

一般に、OOP成分は再使用のできるソフトウェアモジュールであり、それはオブジェクトの型に合ったインターフェースを提供し、成分を集集合化したアーキテクチャーによって実行時にアクセスできる。成分集集合化アーキテクチャーは一連のアーキテクチャーメカニズムであり、違ったプロセスにあるソフトウェアモジュールがお互いの能力や機能を使えるようにしている。これは一般に、アーキテクチャーを構成する共通の成分オブジェクトモデルを想定して実施される。

この点で、オブジェクトとオブジェクトのクラスを分けておくことは意味のあることである。オブジェクトは、よく単にクラスと呼ばれている、オブジェクトのクラスの一つの例である。オブジェクトのクラスは青写真とみることができる。そこから多くのオブジェクトを形成することができる。

OOPによって、プログラマーは他のオブジェクトの一部であるオブジェクトを作り出すことができる。例えば、ピストンエンジンを示しているオブジェクトは、ピストンを示しているオブジェクトと成分関係を持つという。実際、ピストンエンジンは、ピストン、バルブ、その他多くの部品を有している。ピストンがピストンエンジンの一要素であるという事実は、OOPでは2つのオブジェクトを論理的にまた語義的に示すことができるということである。

OOPはまた、他のオブジェクトに「従属」しているオブジェクトを作り出す。2つのオブジェクトがあり、一つがピストンエンジンを示しており、他のものがピストンがセラミックでできているピストンエンジンを示しているとき、その場合この2つのオブジェクトの関係は成分のそれではない。セラミックピストンエンジンはピストンエンジンを作るものではない。それはむしろピストンエンジンの一種であって、ピストンエンジンよりも一つ多く限定を持っている。そのピストンはセラミックでできている。この場合、セラミックピストンエンジンを示しているオブジェクトは派生オブジェクトと呼ばれる、そしてそれはピストンエンジンを示しているオブジェクトの特徴のすべてを引き継いでいて、それにさらなる限定や詳細さを加えている。セラミックピストンエンジンを示しているオブジェクトは、ピストンエンジンを示しているオブジェクトに「従属」している。これらのオブジェクトの関係は継承と呼ばれる。

セラミックピストンエンジンを示しているオブジェクトあるいはクラスが、ピ

ストーンエンジンを示しているオブジェクトの特徴をすべて継承しているとき、それはピストンエンジンクラスに定義されている標準ピストンエンジンの温度特性を引き継いでいる。しかし、セラミックピストンエンジンオブジェクトは、セラミックの特別な温度特性で優れていて、それはメタルピストンのものとはかなり違っている。それは元のを越えて、セラミックピストンの持っている新しい機能を使っている。違った種類のピストンエンジンは違った特性を持つだろうが、それについての同じ基礎になる機能、例えばエンジンに使われるピストンの数、イグニッションの順、潤滑等を持つ。いかなるピストンエンジンオブジェクトにもあるこれらの機能の各々にアクセスするのに、プログラマーは同じ機能を同じ名前と呼ぶだろう。しかし、各種のピストンエンジンは同じ名前の陰にある違ったあるいは優れた実行機能を持つことがある。同じ名前の陰に違った実行機能を隠すこの能力のことを同質異形性と呼んでいて、それはオブジェクト間の連絡を非常に簡単にする。

成分関係、カプセル化、継承、及び同質異形性の概念を用いて、オブジェクトを実世界のすべてのものと同じように示すことができる。事実、オブジェクトオリエンティッドなソフトウェアでオブジェクトとなりうるものの種類を決める上でのみ、実物の論理上の認識は限界がある。いくつかの典型的なカテゴリーは次のようである。

\* オブジェクトは、交通の流れシミュレーションにおける自動車、回路設計プログラムにおける電気部品、経済モデルにおける国、あるいは航空交通管制システムにおける飛行機のような物理的なオブジェクトを示すことができる。

\* オブジェクトは、ウィンドウズ、メニューあるいは図オブジェクトのようなコンピュータユーザ環境の要素を示すことができる。

\* オブジェクトは、人員ファイルや都市の緯度や経度の表のような目録を示すことができる。

\* オブジェクトは、時間、角度、複素数、あるいは面上の点のようにユーザが定義したタイプのデータを示すことができる。

オブジェクトは論理上区別することのできるいかなる状況をも示すことができるという能力があるので、ソフトウェアの開発者はOOPによって、実物のある

特徴をモデル化してコンピュータプログラムを設計し実行することができる。その実物が、物理的な実体であっても、方法であっても、システムであっても、あるいはものの構成部分であっても。オブジェクトによってなにでも示すことができるので、将来より大きなソフトウェアプロジェクトの部分として使うことのできるオブジェクトを、ソフトウェア開発者は生み出すことができる。

ある新しいOOPソフトウェアプログラムの90%が前からあって再使用のできるオブジェクトからなる検証済みで既に存在する部品からできているならば、新しいソフトウェアプロジェクトの残りの10%だけが最初から書かれなければならないだけである。90%がよく検証された再使用のできるオブジェクトの在庫から使えるので、誤りの生じる領域はプログラムの10%だけである。結果として、ソフトウェア開発者はOOPによって前に作ったオブジェクトからオブジェクトを作ることができる。

この方法は複雑な機械を組立品や中間組立品から作るのに非常によく似ている。OOP技術は、それ故に、ソフトウェアエンジニアリングをハードウェアエンジニアリングのようにする。そこでは、開発者がオブジェクトとして利用できる既存の部品からソフトウェアを組み立てる。このことによって、その開発スピードを上げるだけでなく、ソフトウェアの質を高める。

プログラミング言語は、カプセル化や、継承や、同質異形性や、成分関係などのOOP原則を完全にサポートしている。C++言語の到来によって多くの商業的なソフトウェア開発者は、OOPを採用した。C++は、機械が早く実行することのできるコードを提供するOOP言語である。更に、C++は、商業応用にもシステムプログラミングプロジェクトにも適したものである。今までのところ、C++は多くのOOPプログラマーにはもっともよく使われている選択であったが、スモールトークや、コモンリスブオブジェクトシステム(CLOS)及びエイフルのように主要な他のOOP言語がある。加えて、OOPの能力が、パスカルのようなもっと従来からよく使われるコンピュータプログラミング言語にも付け加えることができる。

オブジェクトの利点を次のようにまとめることができる。

\* オブジェクトとその対応するクラスによって、複雑なプログラミングの問題を



より多くの小さく簡単な問題に下げる。

\* カプセル化によって、お互いに関連のつけられる小さな独立したオブジェクトにデータを組織してデータの抽象を行う。カプセル化は、オブジェクト内のデータを偶然の損傷から保護するが、オブジェクト構成員の機能及び構造を呼び出すことで、他のオブジェクトがそのデータと相互作用をすることができるようになる。

\* サブクラシングと継承とによって、そのクラスで使われる標準クラスから新しい種類のオブジェクトを派生させることを通してオブジェクトを拡張及び修正ができる。このようにして、新しい能力を最初から作る必要がなくなる。

\* 同質異形性と多重継承によって、種々のプログラマーが多くの違ったクラスの特性を混合し、合致させることができ、予想していたやり方で関係するオブジェクトと一緒に働くような特別のオブジェクトを生じさせることができる。

\* クラス階層と含有階層によって、実世界のオブジェクトをモデル化して、その間の関係をモデル化する柔軟性のあるメカニズムを生じる。

\* 再使用のできるクラスのライブラリーは、多くの場合に有効であるが、それらにはある限界がある。例えば、

\* 複雑なシステムでは、関係するクラスのクラス階層は何十あるいは何百というクラスがあるので全く混乱するものとなる。

\* コントロールのフロークラスライブラリーの助けを借りて書いたプログラムはコントロールのフローを管理しなければならない。すなわち、それは特別なライブラリーから生み出されたすべてのオブジェクト間の相互作用をコントロールしなければならない。プログラマーはどの機能を、何回、どの種のオブジェクトにするかを決める必要がある。

\* 重複した努力クラスライブラリーを用いてプログラマーは多くの小さなコード部品を使用、再使用できるが、各プログラマーはそれらの部品を違ったやり方で組み合わせることになる。

2人の別のプログラマーが同じ組み合わせのクラスライブラリーを用いて、全く同じことを行うのに、その内部構造、すなわち設計、の全く違った2つのプログラミングを書くのに、各プログラマーがそれぞれのやり方で何百という小さな

意志決定を行っている。必然的に、小さなコード部品は少し違ったやり方で小さなことを行うことになるが、もっとうまくいくはずのものがそのようにはならないことがある。

クラスライブラリーは非常に柔軟性がある。プログラムがより複雑になったときに、更に多くのプログラマーが基本的な問題に基本的な解法を何度も何度も説明せざるを得ない。クラスライブラリーの概念の比較的新しい拡張は、クラスライブラリーのフレームワークを持つことである。このフレームワークは更に複雑であり、小規模パターンと、特定の応用分野での共通の要求と設計を行う大きなメカニズムの両方を持っている共に作用するクラスの大きな集合体である。それらは、パーソナルコンピュータで、メニュー、ウィンドウズ、ダイアログボックス及び他の標準的なユーザインターフェース要素を表示するのに伴う混乱から応用プログラマーを解放するために、当初開発されたものである。

フレームワークはまた、あるプログラマーの書いたコードと他の人の書いたコードの間の相互作用について、プログラマーの思考方法の変化を示す。手続きプログラミングの最初の頃は、プログラマーはオペレーティングシステムから出されるライブラリーを呼び出して仕事を行ったが、基本的にはページの最初から最後までプログラムを実行した。そして、プログラマーは一人でコントロールのフローに責任を持っていた。これは給料小切手を出力したり、数表を計算したり、ただ一つの実行方法しかないプログラムで他の問題を解決するには適したものであった。

図を使ったユーザインターフェースの開発によって、この処理プログラミングのやり方が裏返しになってきた。このインターフェースによって、ユーザがプログラムの論理よりも、プログラムを運転し、ある行為をいつ行うかの意志決定を行うことができるようになってきた。今日、ほとんどのパーソナルコンピュータソフトウェアは、マウス、キーボード、その他のイベントをモニターしているループによってこれを行うようになっていて、ユーザの行う行為によってプログラマーのコードの適当な部分呼び出す。プログラマーはもはや事柄の起こる順序を決める必要がない。代わりに、プログラムは予想できないときに、予想のつかない順序で呼び出される別々の部品に分けられている。このようにして、ユーザ

をコントロールすることをやめることで、開発者ははるかに使いやすいプログラムを作ることができる。それにも関わらず、開発者の書いたプログラムの個々の部品は今でも、ある仕事を行うのに、オペレーティングシステムの提供するライブラリーを呼び出して、イベントループで呼び出された後、プログラマーは各部分にあるコントロールのフローを決定しなければならない。応用コードは今でも一番高いところにある。

たとえイベントループプログラムがプログラマに対して多くのコードを書くことを要求したとしても、各々のアプリケーション毎に別けて書く必要はない。アプリケーションフレームワークのコンセプトは更にループコンセプトを伴う。ベーシックメニュー、ウィンドウを構成する全てのナット及びボルト、ダイアログボックスを処理すること、及びこれらの全てを一体化して働かせることに代えて、アプリケーションフレームワークを用いるプログラマは、アプリケーションコード及び適切なベーシックユーザインタフェースを使用することから開始する。続いて、これらから、彼らはフレームワークの一般的なケーパビリティを意図したプログラムの特有のケーパビリティで置き換えることにより築き上げる。

アプリケーションフレームワークはプログラマが零から書かなければならないであろうコードの総量を縮小させる。しかし、フレームワークはウィンドウを表示し、複写や貼り付けをサポートする等の全く一般的なアプリケーションであるので、プログラマはイベントループプログラムが許容するより多くの程度までの制御を廃することができる。フレームワークのコードは殆ど全てのイベントの取り扱い及び制御の流れを処理し、プログラマのコードはフレームワークがそれを必要とする時にのみコールされる（例えば、適切なデータ構造を作ったり操作するために）。

フレームワークプログラムを書くプログラマはユーザに対する制御を廃するのみでなく、プログラム内のフレームワークに対する制御の詳細な流れをも廃する。このアプローチは、カスタムコードを有し同様の問題に対して繰り返し創作された独立したプログラムに対立するものとして、インタスティングウェイにおいて共同で処理するより複雑なシステムの創作を許す。

従って、以上に述べたように、フレームワークは基本的には、与えられた問題

の領域のための再利用可能な設計の解決法を構成するコオペレーティングクラスの集合である。それは、典型的にはデフォルト動作（例えば、メニュー及びウィンドウ用）を与えるオブジェクトを含み、プログラマは、フレームワークが適切な時刻にアプリケーションコードをコールするために、いくつかのデフォルト動作を継承すること及び他の動作を無効にすることによりそれを用いる。

フレームワークとクラスライブラリとの間には、3つの大きな相違がある。

・プロトコルに対する動作。 クラスライブラリは、プログラムにおける個々の動作を必要とする時にコールすることができる動作の本質的な集合である。一方、フレームワークは、動作のみでなく、フレームワークが与えるものに対してプログラマが与えられると想像するもののための命令を含む、動作が結合され得る方法を決定するプロトコル即ち命令セットを与える。

・無効に対するコール。 クラスライブラリと共に、プログラマがオブジェクトをインスタンス化し、メンバ関数をコールするコード。フレームワーク（例えば、フレームワークをクラスライブラリとして処理するため）と共にオブジェクトをインスタンス化しコールすることが可能であるが、フレームワークの再利用可能な設計の利点を全て活用するために、プログラマは代表的には無効にしフレームワークから呼ばれるコードを書く。フレームワークはそのオブジェクトの間における制御の流れを管理する。プログラムを書くことは、どのようにして異なる部分が共に動作すべきかを特定するよりもフレームワークにコールされる種々のソフトウェアの間における責任を分け合うことを意味する。

・設計に対するインプリメンテーション。 クラスライブラリと共に、プログラマがインプリメンテーションのみを再使用するが、フレームワークと共に彼らは設計を再使用する。フレームワークは、関係するプログラム又はソフトウェアの部分のファミリが動作する方法を具体化する。それは、与えられた領域における種々の特定の問題に適用可能である一般的な設計の解決法を表す。例えば、単一のフレームワークは、たとえ同一のフレームワークと共に生成された2つの異なるユーザインタフェースが全く異なるインタフェースの問題を解決するとしても、ユーザインタフェースが動作する方法を具体化する。

従って、種々の問題及びプログラミング業務に対する解決のためのフレームワ

ークの開発を通じて、ソフトウェアの設計における重要な削減及び開発の努力がなされている。本発明の好ましい実施例は、インターネット上へドキュメントをインプリメントするためのハイパーテキストマークアップランゲジ（HTML）を利用する。HTMLは、あるプラットフォームから他へ持ち運びできるハイパーテキストドキュメントを創作するために用いられる単純なデータフォーマットである。HTMLドキュメントは、広い範囲の領域からの情報を表すために適切である一般的なセマンティックを伴うSGMLドキュメントである。HTMLは1990年から世界的な情報の先駆けであるワールドワイドウェブにおいて使用されている。HTMLは、ISOスタンダード8879：1986、情報処理テキスト及びオフィスシステム；スタンダードジェネライズドマークアップランゲジ（SGML）のアプリケーションである。

遡ると、ウェブの開発ツールは、クライアントからサーバに広がるような及び現存するコンピュータ資源を相互にオペレートするようなダイナミックなウェブアプリケーションを創作するための能力を制限されていた。最近まで、HTMLがウェブに基礎を置く解決法の開発において使用される支配的な技術であった。しかし、HTMLは以下の領域において不十分であることが証明された。即ち、

- ・ 性能が不十分、
- ・ ユーザインタフェースのカーナビリティが制限される、
- ・ 静的にウェブのページを生成することのみ可能、
- ・ 現存するアプリケーション及びデータについての相互的なオペレートの能力の欠如、及び
- ・ スケールに対して無力。

サンマイクロシステムのJava言語はクライアント側の問題の多くを以下によって解決した。即ち、

- ・ クライアント側での性能を改善すること、
- ・ ダイナミックでリアルタイムなウェブアプリケーションの創作を可能とすること、及び
- ・ 広範で多様なユーザインタフェースのコンポーネントを創作する能力を与えること。

J a v aによれば、開発者は強いユーザインタフェース (U I) のコンポーネントを創作することができる。カスタム " widgets" (例えば、リアルタイムのストックティッカーやアニメ化されたアイコン) が創作され得るし、クライアント側の性能が改善される。HTMLとは異なり、J a v aは、性能を改善するためのクライアントへの適切な処理をオフローディングして、クライアント側の有効化の考えをサポートする。ダイナミックでリアルタイムなウェブのページが創作され得る。上述のカスタムU Iコンポーネントを用いることによっても、ダイナミックなウェブのページが創作され得る。

サンマイクロシステムのJ a v a言語は、" インターネットをプログラミングする" ための産業的に認知された言語として出現した。サンマイクロシステムは、J a v aを以下のように定義している。即ち、" 単純で、オブジェクト指向で、分散された、インタプリトされた、強く、安全で、アーキテクチャに対して中立で、ポータブルで、高い性能を持ち、マルチスレッド動作し、ダイナミックで、バズワードにコンプライアントで、汎用のプログラミング言語で、J a v aはプラットフォームから独立したJ a v aアプレットの形態でのインターネットのプログラミングをサポートする。" J a v aアプレットは、開発者がウェブドキュメント (例えば、単純なアニメーション、ページの装飾、基本的なゲーム、等) に対して" インタラクティブなコンテンツ" を付加することを許すサンマイクロシステムのJ a v aアプリケーションプログラミングインタフェース (A P I) に従う、小さくて特殊化されたアプリケーションである。アプレットは、サーバからクライアントへコードを複写することにより、J a v a互換ブラウザ (例えば、ネットスケープナビゲータ) 内において実行する。言語の立場から見ると、J a v aのコアの構成セットはC++に基礎を置く。サンマイクロシステムのJ a v aの文献は、J a v aは基本的には" よりダイナミックなメソッドの解決法のためのオブジェクト的なCからの拡張を伴うC++" であると述べている。

J a v aと類似の機能を与える他の技術は、開発者及びウェブ設計者に対してインターネット及びパーソナルコンピュータのためのダイナミックなコンテンツを構築するための手段を与えるために、マイクロソフト及びアクティブXの技術により与えられる。アクティブXは、アニメーション、3-Dバーチャルリアリ

ティ、ビデオ及び他のマルチメディアのコンテンツを開発するためのツールを含む。このツールはインターネットの基準を使用し、複数のプラットフォーム上で動作し、100以上の会社によってサポートされている。このグループの構築するブロックはアクティブXコントロールと呼ばれ、開発者がハイパーテキストマークアップランゲジ（HTML）に部分的なソフトウェアを埋め込むことを可能とする小さくて速いコンポーネントである。アクティブXコントロールは、マイクロソフトのビジュアルC++、ポーランドのデルフィ、マイクロソフトのビジュアルベーシックプログラミングシステム及び将来におけるマイクロソフトのJavaのためのコードネーム”Jakarta”と言う開発ツールを含む種々のプログラミング言語と共同して動作する。アクティブX技術は、また、開発者がサーバアプリケーションを創作することを許容するアクティブXのサーバフレームワークを含む。この分野の通常の技術者は、本発明を実施するための不適當な実験なしでアクティブXがJavaの代わりに用いられ得ることを容易に認識するであろう。

第1B図はネイティブコードを用いるシステムの好ましい実施例のブロック図である。システムの主要なコンポーネントが第1B図を参照して以下に述べられる。商人ウェブサイト180は、商人の又は他のウェブサーバ通信システムにより与えられるショッピングコモンゲートウェイインタフェース（CGI）190を含む。CGIは、アプリケーションがアクセスし、処理し、入って来るメッセージに応答することを許容するインタフェースである。産業上の用語は、CGIスクリプトを、プログラマがスクリプトな言語におけるウェブサーバアプリケーションを記述する方法として参照する。CGIスクリプトは、ソフトウェアのアプリケーションプログラムの一種である。商人は、代表的には、CGIスクリプトを生成するオーソライゼーションのためのプログラムをセットアップする。オーソリティのサーティフィケートは、また、ウェブサーバを有し、サーティフィケートの要求の取り扱いはCGIスクリプトを利用するであろう。顧客は商人のサイトで買物をするためにこのシステムとインタラクトする。商人システムは、また、ペイウィンドウのヘルパーアプリケーションを使用している顧客から始まる支払いのトランザクションを受け取る。支払い命令192は、顧客が支払いを

選択した時に商人システムにより生成されるマルチパーパスインターネットメールエクステンション (MIME) メッセージを表すファイルである。このMIMEメッセージは、注文の情報と、商人に特有の支払い命令とを含む。注文の情報は、商品及びサービスの注文 (GSO)、送り先、商人のユニフォームリソースロケータ (URL)、商人のサーティフィケート、商人の名前及びアドレス、及び注文に関連させられたトランザクションIDを含む。支払い命令は、商人の受け取る支払い文書と支払いのプロトコルの選択とを含む。

商人システムからのMIMEメッセージを受け取ると、ブラウザは、ブラウザにおいてヘルパーアプリケーションとして規定されるペイウィンドウのアプリケーションを起動する。ブラウザはMIMEメッセージをペイウィンドウに送り、ペイウィンドウは支払いを完了するために顧客とインタラクトする。この分野の通常の技術者は、Java、ネットスケープ/オラクルのプラグイン及びアクティブXが容易にこの実施例のアーキテクチャの基礎をなすMIMEメッセージの代わりになることを容易に理解するであろう。Javaの実施例は、他の好ましい実施例に関連した他の例を与えられるための選択的な実施例として以下に説明される。

アドレス命令ファイル194は、ウォレットデータベースから彼又は彼女のアドレスを自動的に埋めるために顧客が商人ウェブのページ上のボタンをクリックする時に、商人システムにより生成されたMIMEメッセージを表す情報を含む。MIMEメッセージは、商人のURLと、ウォレットデータベースから商人が捜すしているアドレスの型とを含む。ウォレットは顧客の送り先と請求書のアドレスとを含む。一旦MIMEメッセージが商人システムから受け取られると、ブラウザは、ブラウザにおいてヘルパーアプリケーションとして規定されるベリフォン (VeriFone) ウェブサイト184のアドレス徴用アプリケーション176を起動する。アドレス徴用アプリケーション146、176はアドレス徴用要求を完了するために顧客とインタラクトする。

銀行ウェブサイト162でのサーティフィケート発行は銀行ウェブサイト182に帰属する。それは、顧客へのSETコンプライアント/X.500サーティフィケートの発行を利用する。このシステムのインプリメンテーションは1つの



銀行から他について一様ではない。しかし、高いレベルで、このシステムは顧客の個人情報を取捨する。情報を処理した後、システムは顧客への支払い文書に従ってサーティフィケートを発行する。

銀行ウェブサイト182での単一口座のウォレット160は、サーティフィケート発行システムにより生成されたMIMEメッセージを表す。このMIMEメッセージはペリフォンウォレットを含む。ペリフォンウォレットは、単一の支払い文書とこれに関連するサーティフィケートとを含む。安全上の理由から、プライベートキーはウォレットには含まれない。顧客がサーティフィケートを発行した時、このMIMEメッセージはブラウザに送られる。ブラウザは、ブラウザにおいてヘルパーアプリケーションとして規定されるサーティフィケートインストールアプリケーション174、144を起動する。サーティフィケートインストールアプリケーション174、144はMIMEメッセージを読み、ウォレットデータベース158にウォレットをインストールする。

種々のヘルパーアプリケーション198、172、174、176は、以下のヘルパーアプリケーションを含み、顧客の買物の体験を楽にし効率良くするために与えられる。ペイウィンドウヘルパーアプリケーション188は、商人に対する支払いをオーソライズするため、彼らのウォレットを管理するため、彼らの既に完了した支払いのトランザクションを検査するため、及び、ウォレットについての家計の活動を行うため、顧客により利用される。このアプリケーションは顧客のデスクトップ上で”ヘルパー”アプリケーションとして規定される。ブラウザは、商人システムがMIMEメッセージの支払い要求を送る時に、このアプリケーションを起動する。

ペイウィンドウセットアップヘルパーアプリケーション172は、ヘルパーアプリケーション及び他のモジュールをウェブサイトから顧客のデスクトップ上にインストールするために、顧客により利用される。顧客が最初にアプリケーションをインストールしようとする時、顧客はデスクトップ上にヘルパーアプリケーションを持たない。従って、アプリケーションの最初のインストールは、顧客に2つのステップの実行を要求する。最初に、ユーザは彼らのデスクトップにシステムパッケージをダウンロードしなければならず、次に、ユーザはシステ

ムの圧縮を解凍しインストールするためのセットアップを行わなければならない。その後、顧客が新しくリリースされたシステムソフトウェアを購入する度に、ブラウザは、適切な他のシステムモジュールを交替してインストールするこのヘルパーアプリケーションを起動する。

サーティフィケートインストールヘルパーアプリケーション174は、銀行により発行されたウォレットをインストールするために利用される。銀行のサーティフィケート発行ウェブシステムがペリフォンウォレットを含むMIMEメッセージを送る時、ブラウザはこのアプリケーションを起動する。このアプリケーションは、ウォレットに含まれた支払い文書が現存のウォレットに複写されるべきか又は新しいウォレットに維持されるべきかを決定するために、顧客に質問する。このアプリケーションは、その後、支払い文書及びサーティフィケートをウォレットデータベース158にインストールする。

アドレス徴用ヘルパーアプリケーション176は、彼又は彼女のアドレスを特定のウォレットから直接商人システムに送るために、顧客により利用される。商人システムが顧客の送り先及び／又は請求先アドレスを要求MIMEメッセージを送る時、ブラウザはこのアプリケーションを起動する。このアプリケーション176は、ウォレットが既に開いていないでかつ顧客に要求したアドレスが示されていないならば、ウォレットを開くために、顧客に質問する。一旦顧客がアドレスを承認すると、アプリケーションは要求した情報をHTMLフォームで商人に対して”投函する”。このフォームのHTMLフォーマットは標準化されている。この（アドレス徴用）構成を採用する全ての商人システムはこのフォームを受け入れている。加えて、ヘルパーアプリケーションを走らせるために要求される他のソフトウェアモジュールが存在する。最初の頃に、これらのモジュール及びヘルパーアプリケーションは、顧客により彼の又は彼女のデスクトップ上にダウンロードされインストールされる。その後、セットアップヘルパーアプリケーション172が将来リリースされるヘルパーアプリケーション及び他のソフトウェアモジュールを自動的にインストールする。アドレス徴用は実施例として説明されたので、この分野の通常の技術者は、服のサイズ、音楽の好み、母親の名前又は他の商人の助けになる情報もまた類似の手段を通じてアップロードされ得る

ことを、この実施例から離れることなく、容易に理解できるであろう。

第2図はJ a v aに基礎を置く本発明の実施に関連する選択的な好ましい実施例を与える。J a v aは、インターネット及びH T M Lを利用することを可能とするアプリケーションのためのネットワークアプリケーションである。J a v aは、彼らのネットワークアプリケーションの開発の速度を上げるためにサンマイクロシステムズにより開発されたオブジェクト指向言語である。

先に述べた固有の実施例とJ a v a版との間の相違のみが、論を明確にするために述べられる。商人ウェブサイト180における支払い命令アプレット200は、顧客のデスクトップ186上のペイウィンドウアプレット210、230に対する注文情報を配達するための応答が可能である。この注文情報は、ペイウィンドウシステムのビューネイティブコードセクションにおいて述べられている支払い命令のM I M Eメッセージに含まれているのと同じの情報を有する。

前記アプレットは商人により顧客に表示される支払いH T M L 192のページの一部である。前記アプレットは、支払いのページが商人システムG S O、送り先アドレス、商人、商人のサーティフィケート、商人のU R L及びボタンテキストにより生成された時に、以上のパラメータが適切なデータでセットされることを要求する。アプレット200は”ペイ””ペイウィンドウ使用”と呼ばれるボタン、即ち、ボタンテキストパラメータの値を表示する。一旦クリックされると、アプレットは上記の情報を顧客のデスクトップ上のペイウィンドウアプレットに配達する。顧客は、支払いのトランザクションを完了するために、ペイウィンドウ210のアプレットとインタラクトする。

商人ウェブサイト180のアドレス徵用アプレット204は、顧客の送り先及び／又は請求先を得るために、商人システム180により利用される。このアプレットは”Vウオレット”と呼ばれるボタン、即ち、ボタンテキストパラメータの値を表示する。一旦クリックされると、アプレット204は顧客のアドレスを利用してアドレスウィンドウアプレット214を起動する。アドレスウィンドウアプレット214は顧客とインタラクトして、アドレス情報を商人システム180に送る。アドレス情報は、その後、本来のシステムにおける処理と矛盾なく処理される。

銀行ウェブサイト182におけるサートیفিকেイト発行CGIスクリプト162及び単一口座アプレット160は、本来のシステムにおいて述べたように処理される。銀行ウェブサイト182のサートیفিকেイトインストールアプリケーションアプレット220は、顧客のサートیفিকেイトを顧客のデスクトップに配達するために、サートیفিকেイト発行CGIスクリプト162システムにより利用される。

顧客の買物を容易にし効率良くするために、種々のアプレットがウェブサイトで与えられる。これらのヘルパーアプリケーションは、セットアップアプレット212、ペイウィンドウアプレット210及び上述したペイウィンドウヘルパーアプリケーションのネイティブセクションに類似のアドレスウィンドウアプレット214を含む。

第3図は本発明の好ましい実施例に関連するアーキテクチャのブロック図である。アプリケーションのグラフィカルユーザインタフェース（GUI）部分がユニシャライズされる関数ブロック300において、処理が開始される。GUIアプリケーション300は顧客に買物の過程において注文するため及び支払いを行うためのサポートを与える。ウォレットの生成；輸入、サートیفিকেイト及び支払い方法の生成及びメンテナンス；及びトランザクション登録の検査及び報告のために与えられるGUIコンポーネントも存在する。ヘルパーアプリケーション及びアプレットのための表示画面の設計、及びこれに関する論理は以下において個々に詳細に述べられる。

サートیفিকেイトマネージャ304は、銀行からの顧客のサートیفিকেイトの自動的なダウンロード、顧客の及び商人のサートیفিকেイトの有効化、及びサートیفিকেイトの更新の自動的な徴用を管理する。

支払いマネージャ306は商人システムから受け取られた支払い要求を統合し、完了する。支払い要求は、本来のコードのインプリメンテーションにおけるMIMEメッセージを通して、又は、Javaのインプリメンテーションにおけるアプレットを通して、受け取られる。受け取られた支払い要求は、最後のGSO、送り先の名前、商人のサートیفিকেイト、商人のURL、クーポン及び支払いの合計を含む。マネージャ306は、その後、支払いのとトランザクションを

オーソライズし完了するために顧客とインタラクトするため、支払い関連GUIコンポーネントと通信する。マネージャは、また、顧客の支払い命令及び商人の好ましい支払いプロトコルに基づいて支払いのプロトコルを決定するために応答可能である。

マネージャ306は、特定のHTTPサイトに対する支払いを行うための支払いマネージャ306とのインタフェースへのOEMを可能にする、適切に規定されたアプリケーションプログラミングインタフェース(API)を含む。支払いマネージャ306に関連する詳細なロジックは第4図に表される。

支払いマネージャ306は支払いプロセスにおける標準的なオペレーションを実施する。例えば、一旦支払いが完了すると、レシート及びトランザクションは自動的にウォレットファイルに転送される。好ましい実施例に関連する支払いマネージャのアーキテクチャは第4図に表される。ユーザは、種々のトランザクション410、408、406、404及び402に応答しこれらを送るインタフェース400を通して、支払いマネージャ430とインタフェースする。前記トランザクションは次のレコード、支払いレコード、レシート、支払い文書のアクセプタンス及びGSOコンポーネントを含む。

次に、支払いマネージャ430はトランザクション414及びレシート420をウォレットマネージャ422に送り、支払い文書、サーティフィケート及びプライベートキーをウォレットマネージャ422から受け取る。

支払いマネージャ430は、また、商人の氏メッセージ460、顧客のサーティフィケート及びPKハンドル450、商人のURL442、支払い440、単一のレシート434及びGSO、選択された支払いプロトコル及び選択された支払い文書432を含むトランザクションを、プロトコルマネージャ470に送りまた受け取る。支払いマネージャ430は、更に、関数ブロック480に示すように、支払いアプレットからの入力又は商人からのMIMEメッセージを受け取る。支払い処理の1つの状況は、支払いプロトコルを単一のAPIのカプセルに入れる顧客支払いクラスライブラリ(CPCL)である。支払いプロトコルをカプセルに入れることにより、アプリケーションはプロトコルの多様性から隔離される。SETプロトコルは、セキュアエレクトロニクストランザクション(SET

）プロトコルのクライアント側のコンポーネントのインプリメンテーションを与える。サイバーキャッシュのマクロペイメントプロトコルのクライアント側のコンポーネントの完全なインプリメンテーションもまた与えられる。

ウォレットマネージャ4.2.2はウォレットに対して標準的なインタフェースを与える。それは、ウォレットデータベースの構造及び支払い文書のデータ構造を決定し、ウォレットへのアクセスを制御し、1つ以上のアプリケーションが同一のウォレットを開こうとしたかの並列的なチェックを与える。ウォレットマネージャ4.2.2へのインタフェースは、ウォレットマネージャとのインタフェース及びウォレットデータベースへのアクセスのためのOEMを許容すると公開されている。

ウォレットマネージャは以下のサブコンポーネントからなる。即ち、ウォレットアクセス。このコンポーネントはウォレット情報を読み書きするためのインタフェースを与える。

トランザクションマネージャ。このコンポーネントは、ウォレットデータベースへのウォレットに対応するトランザクションを読み書きするためのインタフェースを与える。

支払い文書マネージャ。このコンポーネントは、特定の支払い文書アクセスコンポーネントへのコモンインタフェースを与える。

クレジットカードアクセス、デビットカードアクセス、チェックアクセス。これらのコンポーネントは特定の支払い文書进行处理する。

データマネージャは一般的なデータ項目及びデータベースのレコードの記憶及び回復を与える。データフィールド、インデックスフィールド又は全体のデータレコードはエンクリプトなもの（`encrypted`）としてマークされ、エンクリプションプロセスは広く自動化されている。データベースマネージャは、異なる支払い方法に適したデータベースのレコードについての特定の知識を持たない。このレイヤーは、新しい支払いの方法が導入された時に、要求される変更を小さくするように、分離されている。しかし、RSAキーの対及びサートイフィケイトは”単一の”データ形式として考慮されるであろう。このコンポーネントは、また、コンピュータのディスク上の又はスマートカードに含まれたウォレッ

トファイルをサポートするアブストラクションを与える。

オープンデータベースコネクティビティ (ODBC) / J a v a データベースコネクティビティ (JDBC) のコンポーネントは、正式なデータベースコンポーネントが要求されるデータベースコネクティビティを与える。スマートカードウォレットの実施例は、ウォレットデータがクリプトグラフィック (c r y p t o g r a p h i c) トークンにより蓄積され及び／又は守られることを許容する。

好ましい実施例は、個人的な情報及び好ましいインプリメンテーションのマルチプルな支払い方法についての情報を含む”ウォレット”からなる単一のファイル又はファイルのディレクトリを含む。これらの支払い方法 (ビサカード、クレジットカード、スマートカード、マイクロペイメント等) は、また、口座番号、サーティフィケート、キーの対、満期データ等のような情報を含む。ウォレットは、ウォレットを用いて行われた支払いの各々に関係する全てのレシート及びトランザクションレコードをも含むと予想される。クリプトグラフィック A P I コンポーネントは、R S A 及び関連するクリプトグラフィックソフトウェア又はハードウェアのための標準的なインタフェースを与える。このサポートは、エンクリプション、シグネチャ、及びキーの生成を含む。キー交換アルゴリズムの選択、シンメトリックエンクリプションアルゴリズム、及びシグネチャアルゴリズムは、全てコンフィギュラブルであるべきである。基礎のクラスは一般的な動作を規定し、派生したクラスは種々のセマンティックオプション (例えば、クリプトグラフィに基づくハードウェアに対するクリプトグラフィに基づくソフトウェア) を取り扱う。

クリプトグラフィックなソフトウェアの部分は R S A 及び D E S のサポートを与える。これは、特定の顧客のための選択的なインプリメンテーションに依存する S U N、R S A 又はマイクロソフトのシステムのコンポーネントを利用することを与えるであろう。クリプトグラフィックなハードウェアは、クリプトグラフィ A P I の基礎を支え、かつ、クリプトグラフィエンジンのシェルフでクリプトグラフィソフトウェアを置換するために利用することが可能な低いレベルの A P I を生成する。

メッセージシーケンスのチャートは、顧客、ブラウザ及び／又はスメル（Sumeru）システムの種々の主コンポーネントの間のメッセージ／データの流れを述べる。システムの主コンポーネントは、vPOS、ペイウィンドウ、及び支払いゲートウェイを含む商人システムである。商人システムは、顧客が買物をし、ペイウィンドウアプリケーションにより送られた支払いトランザクションを受け取り、及び取得した銀行へ支払いトランザクションを送ることを許容する。顧客支払いクラスライブラリ（CPC L）モジュールは、支払いトランザクションを安全に顧客から商人に送るアプリケーション内のレイヤーである。

第5図は本発明の好ましい実施例に関連する顧客支払いメッセージシーケンスの図である。この図は、顧客、ブラウザ、商人システム、ペイウィンドウアプリケーション、及びCPC Lの間におけるメッセージの流れを表す。このメッセージの流れは、注文が完了して顧客が支払いを選択した時から、支払いが証明されてレシートが顧客に戻される時までの支払いのプロセスを述べる。

ペイウィンドウアプリケーションの本来のインプリメンテーションとJavaのインプリメンテーションとの間の相違は、ペイウィンドウへの注文情報の配達にある。一旦注文情報がペイウィンドウにより受け取られると、メッセージ／データの流れは双方のインプリメンテーションで同一である。本来のインプリメンテーションの場合、注文情報はMIMEメッセージを通して配達される。このMIMEメッセージはドキュメントファイルを通してブラウザによりペイウィンドウへ送られる。Javaのインプリメンテーションの場合、注文情報はアプレットを通してペイウィンドウへ配達される。商人システムは、注文を次にペイウィンドウへ配達するブラウザへ注文情報と共にアプレットを送る。一旦注文が受け取られると、ペイウィンドウは顧客及び支払いプロセスの完了のためのプロトコルモジュールとインタラクトする。

#### 注文入力と注文計算のクリック520

このメッセージは消費者の注文入力と‘注文計算’ボタンをクリックすることを表す。消費者のショッピング体験は、支払いプロセスを高輝度にする目的として全てこの一つのメッセージフローに集約される。実際になされるショッピングプロセスはさまざまであるが、注文を生成する目的は変わらない。



## 注文530

このメッセージはHTML形式によりブラウザにより商店に送られる注文情報を表す。

G S O, P P P, A I, 商店の証明およびURLを伴うPayment Applet 540

注文を受け取ると、商店システムは支払いの額を計算する。このメッセージは、G S O, P P P, A I、商店証明およびURLを含むジャバのPayment アプレットにより支払い額を明細化する商店システムにより送られるHTMLページを開く。

Payment Appletを実行する545

ジャバイネーブルブラウザは、Payment アプレットを実行する。このアプレットは、消費者がクリックするように“Pay”と呼ばれるボタンを表示する。これは、商店により提供されるHTMLページに埋め込まれている。

Payをクリックする550

このメッセージは、支払いの額を確認した後に消費者によりブラウザにPay ボタンのクリックを表す。

G S O, P P P, A I, 商店の証明およびURL 560

このメッセージはジャバアプレットにより搬送されるG S O, P P P, A I, 商店の証明およびURLを表す。ここで、ジャバアプレットはPayWindow アプリケーションにそれらを引き渡す。

商店証明562

このメッセージは商店の有効性をチェックするためのC P C Lモジュールに送られる商店の証明を表す。

商店の有効性564

C P C Lモジュールは商店の証明を確認し、そしてこのメッセージを商店が有効な商店であるかないかを示すPayWindowにこのメッセージを送る。

Wallet, Payment装置566

このメッセージは消費者に表示されるウォレット（書類入れ）および支払い装置を表す。ウォレットからの全ての支払い装置が消費者に示さることはない。商

店により受け入れられたものだけが示されるだけである。

#### Payment 装置 568

このメッセージは、消費者により選択される支払い装置を示す。このメッセージは“Select Payment Method”ウィンドウの支払い画像上でユーザがダブルクリックした時にその時のデザインで作成される。

#### GSO 570

これは、“Make Payment Authorization”スクリーンにおいてGSOが消費者に表示されることを指示する。

#### Payment の Authorization 572

このメッセージは、消費者による支払いのオーソライゼーションを表す。消費者は、“Payment Authorization”スクリーンで“Applet”ボタンをクリックすることにより支払いを認める。

#### Payment Protocol を決める 574

消費者が支払いを認めると、支払いプロトコルが商店のPayment Protocol Preferenceと消費者が選択した支払い装置に基づいてPay Windowにより決められる。

#### Payment Authorization 575

これらのメッセージは、商店のURL、GSO、使用する支払いプロトコル（PP）、口座番号、証明書およびプロトコルモジュールに送られる支払い装置に関連した個人キーハンドル（PK）を表す。

#### Payment Authorization を伴う GSO 576

このメッセージは、商店システムにプロトコルモジュールにより送られる支払い装置を示す。このGSO、PI、消費者証明およびPKは支払いプロトコルに基づいてパッケージされる。

#### 署名された領収書 578

このメッセージは、商店からのプロトコルモジュールにより受領されたデジタル的に署名された送信領収書を示す。

#### ハッシュされた値をもつ安全な領収書 580

デジタル的にサインされた取引領収書は、将来の参照のためにPayWind

owによりセーブされる。

支払い成功582

これは、取引領収書と‘Payment Successful’が消費者に表示されることを表す。

望ましい実施例に従ったPayWindowの必要条件

支払いアプリケーションについての消費者の要求には、特定のターゲットのハードウェアプラットフォームをサポートすること、公衆ネットワーク上での支払い情報の通信、簡単な構成とアクセス可能性、簡単化された注文入力と取引の遂行を保証すること、安全に確実に支払い情報を記憶すること、幅広い多様な支払い装置を提供すること、消費者の好みの店のURLを持つこと、店を訪れるために支払いウィンドウからブラウザを開くこと、レコードを保持し、そして取引データに対するレコードを出力すること、ショッピングのアシスタンスを提供すること、一個のアプリケーションしかない多数のユーザをサポートすること、特定の商店に受入られる商店の証明と支払い形式表示すること、デジタルの資格認定を扱うこと、単一の支払い形態しかない多数の口座を提供すること、商店値の付加もしくはロイヤルティプログラムをサポートすること、およびサポートは他のクライアントの装置に移動可能でなければならないことのような様々な必要な条件がある。

一個の商店は、支払いアプリケーションに対してセットされた異なる必要条件を持ち、それは矛盾のない情報フォーマット、ロイヤルティとブランド化の機会、消費者が選択する多数の支払いタイプ、商店の記憶装置としっかり一体であること、システムから商店の記憶装置にリンクすること、システムを利用するリスク管理と割合のより良いことを含む。リンクは2つのディスプレイの間のハイパーテキスト結合、シングルディスプレイにおけるエリア間のポイントツーポイント接続、ユニフォーム・リソース・ロケータ（URL）、インデックス、ある実行エリアから他の実行エリアに制御を移すことの容易なアドレスもしくは他の手段を参照することである（機械、ネットワーク・ロケーションもしくは人工衛生リンケージおよびマイクロウェブ送信においてさえ交差して）。

この発明に従う望ましい実施例は、電子支払いをなす消費者により使用される

ソフトウェアアプリケーションを提供する。それは消費者のワークステーションで実行され、そしてWWWブラウザへのフリー・スタンディング・アプリケーション、もしくは消費者アプリケーションに埋め込まれているのいずれかであり、そして消費者が特定のアイテムに対する注文プロセスを実行した時にはいつも要求され、そして支払う用意をするものである。望ましい実施例に従うPay Windowアプリケーションは、多数のローカルユーザをサポートする。各ユーザのデータは、独立に記憶されそして安全にされている。プログラムのスタートアップにおいて、ユーザはIDとパスワードにより署名しなければならない。あらゆる情報（ローカルディスクもしくはフロッピー上での）は、プログラムが要請される度にユーザにより入力されたパスワードで暗号化される。

#### Payment装置

社会的、経済的および文化的ファクタに関係する様々な大きさの取引および異なる支払い装置に従って、多数の支払い装置に対してサポートが要求される。今日多くあるように、商人から購入する消費者はその個人の好みに依存した支払い方法の選択を持たねばならない。そのサポートされる支払いは、クレジットカード、電子チェック、電子マネー、マイクロペイメント（電子コイン）、請求カードおよびスマートカードを含む。

#### 安全な支払いプロトコル

望ましい実施例に従う核となる特徴の一つは、安全性と信頼性のある方法で支払いを受けることおよびなす能力である。インターネットにおける安全な支払いは、本発明の他の概念に従って支払い解決をする今日のPoint of Sale（POS）ターミナルと同じように安全で信頼できるようになっている。安全性のプロトコルの選択は、技術設計的配慮で考慮できる一方、その解決を素早くマーケットに供することにおいて高い市場価値がある。

#### 支払いプロセス

支払いプロセスは、管理的機能から切り離されている。消費者が商店から購入することを決めると、依頼人はユーザ名とウォレットパスワード、そして商店および注文情報を表示し、ユーザがウォレットから支払い装置を選択し、そして注文容認とデジタル領収書を表示して捕獲する。基本的に支払い装置として、メン

バーシップ、識別性、証明もしくはオーソリティの証拠であるなんらかの装置を利用できる。例えば、ビデオメンバーシップのためのカードがインターネットを介して電子的にビデオをチェックアウトするのに利用される。

#### サポートされている取引

望ましい実施例に従うアプリケーションは、セールを生成することおよび取引タイプを書き換えることができる。それはまた、ユーザの望みのファイルに基づいてアドレスの自動設定 (fill-in) を提供し、そしてもし、取引が開始された時にいかなる情報も明細化されないなら、最終支払い取引から情報が不行情報として利用される。

#### ショッピングサポート (ショッピングコンパニオン)

望ましい実施例に従うアプリケーションは、また、商店および消費者が訪問のリンケージの前によくかよった商店からのURLをもつ支払いとショッピング情報を共有する。

#### 管理機能

アプリケーションの利用を簡単にするために、実際の支払いと管理的機能が独立している。ユーザはオンラインの支払いプロセスの実行を試みている間に管理情報を示されない。それ故に、次の管理的機能がサポートされる。

#### 日付管理

日付は、ユーザの要求により基本的に周期的にディスクに自動的にバックアップされる。ユーザはさまざまな方法でバックアップの内容を見ることができる。例えば、ユーザは、バックアップ、記憶されたあらゆるデータ、選択されたユーザ情報のみ、もしくは支払い履歴情報以外のあらゆるものから選択して再記憶できる。データを出力は、またeメールに対する選択されたデジタル領収書もしくは注文確認に対してサポートされ、もしくは銀行もしくは他の商店との論争解決を印刷することをサポートする。データ出力は、会計プログラムである *Quicken*, *Microsoft Money* and *Managing your money* に提供される。

#### 領収書と取引履歴管理

領収書と取引履歴管理はまた、時間、支払い装置タイプ、商店による、および

遂行された、遂行されなかったもしくはペンディングの取引による注文履歴を見るために提供される。ページ注文履歴取引がまた時間もしくは支払い装置タイプにより注文履歴ファイルをページするために提供される。クレジットカード番号、満了日、名前、小切手アドレス、公衆キー証明を含むなされた注文に対するデジタル領収書を示す他のレポートが提供される。注文確認は、またなされたあらゆる注文に対して送信され、そしてもしアイテムが外部会計管理プログラムに出力されるなら、出力ステータスタグが生成される。

#### ユーザ管理

さまざまなユーザ管理装置が提供される。それらは、新しいユーザが生成される時に、ユーザに対する空のウォレットを作り、そしてユーザに対してデフォルトパスワードをセットする特徴を含む。削除ユーザ、変更ユーザパスワード、およびユーザとしての署名がまた提供され、しかしユーザのパスワードが活性化されることを要求する。ユーザのビューストがまた提供され、そしてシステムに定義されたユーザを調べるためにパスワードは必要としない。

#### ウォレット管理

ウォレットはユーザ特定であり、そしてウォレットが有効に属するユーザに特定である必要がある。望ましい実施例は新しい支払い装置を付加するため、支払い装置を削除するため、支払い装置のリストと支払い装置を修正するための処理を提供する。

#### L o y a l t y   P r o g r a m   C o u p o n

商店が使用できる他のロイヤルティプログラムと同様にクーポンがまた受け入れられ、そして装置が、ロイヤルティプログラム状態を見るため、そして存在するクーポンを見るために提供される。

#### 共通動作性

本発明の望ましい実施例は、多くのブラウザと商店システムと共通的に動作する。ブラウザおよび商人に交差する共通動作性の要求はベースレベル機能に適用される。

#### スマートカード統合

スマートカードの統合は、消費者に他の支払い方法を提供し（スマートカードの

記憶値)、ウォレット情報の携帯性(キー、支払い、支払い情報)。

国際的ローカライゼーションの必要条件

特定の国へのローカライゼーションに対するサポートは、多数の世界中の国向けの製品に構築される。

第6図は、本発明の望ましい実施例に応じた支払いユーザインタフェースの詳細ロジックを説明するフローチャートである。処理は機能ブロック600から開始し、その機能ブロックはジャバアプレットとして埋め込まれたインターネットブラウザが支払いボタンの選択を検出するものであり、そしてテストが、ウォレットが開かれているかどうかを決定するために決定ブロック610で実行される。もし、そうなら、その時、支払い方法が機能ブロック630で決定され、そしてもしユーザが処理をキャンセルするなら、その時632で制御がインターネットブラウザに戻される。もし、ユーザが機能ブロック634で支払いオーソライゼーションを選択したら、その時、ユーザが機能ブロック650で示されるようにオーソライズをキャンセルできるなら、機能ブロック640で示されるように支払いオーソライズを受け入れ、もしくは機能ブロック630へ制御を移動する635で示されるように支払い方法を変更する。もし、テストが決定ブロック610で実行された時、ウォレットが開いていなかったら、その時ウォレットは、機能ブロック620で示されるように開かれ、そして制御が第7図で示される論理と一致する処理のために622でAdministration Screenに渡るか、もしくはもしユーザが処理をキャンセルするなら、その時制御は624で示されるインターネットブラウザに戻される。

第7図は、望ましい実施例に従う支払いウィンドウの管理モードに関連する詳細論理のフローチャートである。アプリケーションがデスクトップに置かれた時、処理が機能ブロック700で始まる。ユーザは、多くの利用可能な選択をもつPay Window 710を最初に見る。もし、ユーザがウォレット720を選択したなら、その時機能ブロック721でウォレットが開かれ、そして制御が支払いの特定の様式が選択されるように機能ブロック722に渡され。支払いの特定の様式が724で選択されたら、もしくは支払い選択が726でキャンセルされたら、その時、制御が機能ブロック720でメインウィンドウに戻される。し

かし、もし、ユーザが機能ブロック721で新しいウォレットを開くことを選択したなら、その時、さらに処理するために、制御は管理機能ブロック740に渡される。

もし、ユーザがメインウィンドウ710から管理モードを選択したら、その時他のテストが、ウォレットが開かれているかどうかを決定するために決定ブロック732で実行される。もし、ウォレットが開かれているなら、その時、さらに処理するために管理機能ブロック740に渡される。もし、ユーザがメイン支払いウィンドウ710からレジスタ処理712を選択するなら、その時、800で第8図に詳細に示されるレジスタ機能ブロックに渡される。もし、ユーザがレポート処理を選択するなら、その時、さらに処理するために、制御が第8図のレポート機能ブロックと860に渡される。もし、ユーザがバックアップ／再記憶716を選択するなら、その時バックアップもしくは再記憶が、上記のような望ましい実施例に従ってオペレーティングシステムツールを利用して実行される。

もし、ウォレットが決定ブロック732で開かれずに決定されたら、その時、制御がウォレットを開くように機能ブロック734に渡される。ウォレットが開かれた後、それから制御が支払いを選択するように機能ブロック736に渡されるか、もしくはユーザが操作をキャンセルするなら、制御がメインウィンドウ710に渡され、もしくはこれが新しいウォレットであるなら、その時制御が管理機能ブロックに740に渡される。

制御が管理機能ブロックに渡される時、その時、もしウォレットタブが選択されたら、その時制御は機能ブロック750で管理ウォレットページに渡り、そしてもしユーザがブラウジングに関心があるなら、その時、ファイルオープンダイアログが機能ブロック760で開始され、もしくは処理からウォレットを移動する要求が受け取られたら、移動確認ダイアログが機能ブロック770で開始される。もし、ユーザが管理優先を変更することに関心があるなら、その時制御が機能ブロック780に渡される。もし、支払い管理処理が選択されたら、その時、機能ブロック796に示されるように証明がインストールされる機能ブロック790で示されるように管理支払いページに渡される。確認ダイアログが機能ブロック794で付加できるか、もしくは移動確認ダイアログが機能ブロック7



92で処理される。管理処理がなされた時、制御が機能ブロック710のメインウィンドウに戻される。第8図は、望ましい実施例に従うレジスタおよびレポート処理に関連する管理モード処理のフローチャートである。もし管理レジスタタスクが処理されるなら、制御が通過する800で処理が始まる。レジスタ機能は、機能ブロック802で理解され、そしてもし詳細機能が処理されるなら、制御が、商店サイトへのホットリンクが望まれるかどうかを決定するために機能ブロック850に渡される。もし、そうなら、その時、制御がブラウザを起動しそして856で商店サイトに行くために、852を介して渡される。もし、そうでないなら、その時、制御がレジスタ機能802に戻り858を介して渡される。もし、見いだした機能が処理されるなら、その時、制御が、見いだした操作が特定のアイテム844に対して実行され、もしくは見いだした次の操作が842で実行される機能ブロック840に渡される。

見いだされた処理が達成されたかあるいは846でキャンセルされた時、制御がメインレジスタスクリーンに戻される。もし、カスタマイズレジスタ処理が望まれるなら、その時、制御がレジスタをカスタマイズするために機能ブロック830に渡される。もし、クリア操作が必要なら、その時、制御がレジスタをクリアするために832に渡される。カスタマイズ操作が達成される時、その時、制御がメインレジスタ処理に戻るように渡される。もし、レジスタ削除の操作が必要なら、その時、制御が機能ブロック820に渡され、そして削除確認ダイアログが機能ブロック822で処理される。最終的に、もし、出力レジスタ操作が要求されるなら、その時、制御が機能ブロック810に渡され、そしてファイルオープンダイアログが機能ブロック812で開始される。ダイアログが達成される時、その時制御が機能ブロック802に戻される。

もし、レポートボタンが管理支払いスクリーンで押されたなら、その時、機能ブロック880でレポートをカスタマイズするために、882でレポートをクリアするために、もしくは機能ブロック874でファイルオープンダイアログを介して達成されるレポートを872で出力するために、制御がレポート機能ブロック870に860で渡される。

Pay Window

第9図は、望ましい実施例に従う支払いウィンドウ表示の実施例である。支払いウィンドウ980は、適切なパスワードが入力点930を介して入力されたらユーザにパースビットイメージ900もしくはウォレットビットイメージ910、920のようなグラフィック様式の利用可能な支払い装置ホルダーのリストを示す。スクロールバー922は、スクロールされるべき利用可能な支払い方法を許容するように提供され、もしくは上向き矢印924もしくは下向き矢印926を選択することが利用可能な支払い方法を調べるために使用できる。加えるに、970を開くため、新しいウォレット960を作るため、そして支払い950をキャンセルためにスクリーンのボタンで提供されるエリアがある。キャンセルボタン950は、支払いキャンセルスクリーンの表示を要請する。最終的にメッセージ領域は、システムメッセージ940に通信するためにスクリーンのボタンを提供される。

もし、ユーザがマウスボタンの右か左を押せば、ウィンドウの焦点はそれに応じてシフトする。CPUに接続されたキーボードのタブキーは、ディスプレイ上で様々な制御の間にハイライトを動かしたりもしくは循環するのに利用される。もし、入力キーが押されたら、ハイライトされた制御が活性化される。例えば、もし、情報がパスワードフィールド930に入力された後に入力キーが押されたら、その時、パスワードは、有効なパスワードが入力されたかどうかを決定するためにパスワードファイルでチェックする。ウォレットのリストボックスは、システムにインストールされたあらゆるウォレットを示すシングル選択リストボックスである。ユーザは、開くべきウォレットを選択するのにマウスもしくはキーボードを利用出来る。新しいボタン960は、新しいウォレットファイルを作り、そしてウォレット情報入力のために‘Administration’を表示する。

#### 支払い装置

第10図は、好適具体例に従い、ユーザが選択するためのビットマップイメージとして表される支払い装置のリストを表している。支払い装置は、クレジットカード、キャッシュカード、スマートカード、電子キャッシュ、マイクロペイメント、及び電子小切手を含んでいる。オープンボタン970は、支払い装置ス

クリーンを呼び出す。セレクトボタン1000をクリックすることにより、命令の詳細が表示される認可スクリーンに制御は移る。第10図に表された支払い装置は、ユーザの実際のカードに事実上同一のグラフィックイメージ、支払い装置保持者ネーム、装置ネーム、会員期間及び終了日のような情報を持つバンク・アメリカード・VISA1040を含んでいる。このようなイメージは、ユーザのカードをスキャニングし或いはメーカのテンプレートを利用しそしてユーザネームを表すテキストを付加することにより得ることができる。チェブロン当座勘定1020はまた、好適具体例においてサポートされている別のタイプの支払い装置であるユーザの当座勘定を表す拡大チェブロンロゴを備えている。ウエルズ・ファルゴ・VISA1030支払い装置がまた、好適具体例に従い提供される。実際のVISAカードの実際的な表示が第10図に表されているが、当業者は、他のユーザ限定のグラフィック又はテキスト情報が支払い装置を表すために利用することができるということを容易に認識するであろう。キャンセルボタン1010を押すと、支払いキャンセルスクリーンに戻る。ボブのウォレット1050を選択すると、ボブのウォレットパスワード930を入力するために開いたウォレットの表示スクリーンに戻る。いずれかの支払い装置をダブルクリックすると、その特別の支払い装置のための認可スクリーンに入る。支払い装置アイコンをクリックすると、その周りに境界を付して、それが選択されたことを示す。この境界の一例が、ジェーンの財布1060に対して示され、かつジェーンの財布と関連したグラフィックは、1060で示されるような開いた位置におけるグラフィックである開いた状態の証印を有している。同様に、もし、ボブのウォレット1050のような一つのウォレットが選択されるならば、このウォレットと関連したグラフィックは、色の変更、又は、開いたウォレットを描くグラフィックのようなオープンの証印によって表すことができるであろう。

好適具体例に従うソースコードが、詳細なロジック開示の別の形態として以下に提供される。WalletBrowserPanelクラスは、そこに含まれるウォレット及び支払い装置を表示する。それは、PasswordEntryPanel、ImageSelectorPanel及びImageItemクラスを利用する。PasswordEntryPanelクラスは、ユーザからの与えられたウォレットのためのパスワードを受け取る。ImageSelectorPanel

クラスは、ウォレットアイコン及び支払い装置アイコンを表示し、かつユーザにウォレット又は装置を選択／選択解除させる。スクリーン上のアイコンの表示は、**ImageItem**クラスによって管理されている。

```

-----
import java.awt.*;
import java.net.*;
import java.util.*;
public class WalletBrowserPanel extends Panel {
    private PayWindow payWindow;
    private WalletManager walletManager;
    private ImageSelectorPanel walletPanel, instrumentPanel;
    private Button btnOK, btnCancel;
    private Panel buttonPanel, walletDetail;
    private PasswordEntryPanel passwordEntryPanel;

    private Insets insets = new Insets(15, 15, 15, 15);
    private ImageItemRef openWalletImageItem = null;

    public WalletBrowserPanel(PayWindow pw) {

        payWindow = pw;
        walletManager = WalletManager.getWalletManager();
        btnOK = new Button("Open");
        btnCancel = new Button("Cancel");
        buttonPanel = new Panel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER,
15, 10));
        buttonPanel.add(btnOK);
        buttonPanel.add(btnCancel);
        instrumentPanel = new ImageSelectorPanel(false, null);
        walletPanel = new ImageSelectorPanel(false, null);

        Enumeration walletNames = walletManager.getWalletNames();
        String name;
        Wallet wallet;
        while(walletNames.hasMoreElements()) {
            name = (String)walletNames.nextElement();
            wallet = walletManager.getWallet(name);
            try {
                walletPanel.addImage(name, new
                    URL(walletManager.getImageDir() +
                        wallet.getCloseBitmap()), wallet);
            } catch (Exception e) {
                System.out.println(e);
            }
        }
        passwordEntryPanel = new PasswordEntryPanel(this);
        passwordEntryPanel.setPanelSize(new Dimension(175, 290));
        walletDetail = new Panel();
        walletDetail.setLayout(new CardLayout());
        walletDetail.add("PasswordEntryPanel",
passwordEntryPanel);
        walletDetail.add("InstrumentSelection", instrumentPanel);
        setLayout(new BorderLayout());
        add("South", buttonPanel);
        add("West", walletPanel);
        add("Center", walletDetail);
        walletNames = walletManager.getWalletNames();
        name = (String)walletNames.nextElement();
        passwordEntryPanel.setWallet(name);
        displayOpenWallet();
    }
}

```

```

    }
    public void openNewWallet() {
        // ImageItemRef imageItemRef = walletPanel.
        //     getSelectedItem();
        // passwordEntryPanel.setWallet(imageItemRef.getName());
        // btnOK.setLabel("Open");
        // ((CardLayout)walletDetail.getLayout()).
        //     show(walletDetail, "PasswordEntryPanel");
        // payWindow.setTitle("PayWindow - Open Wallet");
        // displayOpenWallet();
    }
    public void getPaymentInstrument() {
        ImageItemRef imageItemRef = walletPanel.
            getSelectedItem();
        Wallet openWallet = payWindow.getOpenWallet();
        Wallet selectedWallet =
            (Wallet) imageItemRef.getUserObject();
        if ((selectedWallet != null) && (openWallet != null) &&
            openWallet.equals(selectedWallet)) {
            // btnOK.setLabel("Select");
            // ((CardLayout)walletDetail.getLayout()).
            //     show(walletDetail, "InstrumentSelection");
            //
            // payWindow.setTitle("PayWindow Choose Payment
            Method");
            displayChoosePaymentMethod();
        }
        else {
            displayOpenWallet();
        }
    }
    public void disableOpenButton() {
        btnOK.disable();
    }
    public void enableOpenButton() {
        btnOK.enable();
    }
    public void newWallet(Wallet wallet) {
        System.out.println("Wallet Browser - newWallet");
        try {
            walletPanel.addImage(wallet.getName(),
                new URL(walletManager.getImageDir() +
                    wallet.getCLOSEBitmap()),
                    wallet);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public Insets insets() {
        return insets;
    }
    public boolean handleEvent(Event evt) {
        if (evt.target == instrumentPanel) {

```

```

        if (evt.id ==
ImageSelectorPanel.EVENT_IMAGEDOUBLECLICK) {
            PaymentInstrument instrument;
            ImageItemRef imageItemRef =
(ImageItemRef) evt.arg;
                instrument = (PaymentInstrument)
                    imageItemRef.getUserObject();
                payWindow.authorizePayment(instrument);
            }
        else
            if (evt.target == walletPanel) {
                if (evt.id ==
ImageSelectorPanel.EVENT_IMAGESELECTION) {
                    ImageItemRef imageItemRef =
(ImageItemRef) evt.arg;
                    Wallet wallet =
(Wallet) imageItemRef.getUserObject();
                    if (wallet == payWindow.getOpenWallet() &&
                        payWindow.isPaymentRequest()) {
                        btnOK.setLabel("Select");
                        ((CardLayout) walletDetail.getLayout()).
show(walletDetail,
"InstrumentSelection");
                        payWindow.setTitle("PayWindow - Choose
                            Payment
Method");
                        displayChoosePaymentMethod();
                    }
                    else {
// passwordEntryPanel.setWallet(imageItemRef.getName());
// btnOK.setLabel("Open");
// ((CardLayout) walletDetail.getLayout()).
// show(walletDetail,
"PasswordEntryPanel");
// payWindow.setTitle("PayWindow-Open
Wallet");
                        displayOpenWallet();
                    }
                }
            else
                if (evt.target == btnOK) {
                    if (evt.id == Event.ACTION_EVENT) {
                        if (btnOK.getLabel() == "Open") {
                            System.out.println("Open");
                            ImageItemRef imageItem = walletPanel.
                                getSelectedItem();
                            Wallet selectedWallet =
(Wallet) imageItem.getUserObject();
// validate the password
// close any open other open wallet
if (openWalletImageItem != null) {
                    Wallet prevOpenWallet = (Wallet)

```

```

openWalletImageItem.getUserObject();
prevOpenWallet.close();

try {

walletPanel.changeImage(openWalletImageItem,
                        new
URL(walletManager.getImageDir() +
prevOpenWallet.getCloseBitmap()));
} catch (Exception e) {
    System.out.println(e);
}

// open selected wallet

walletManager.openWallet(selectedWallet.getName(), "");

payWindow.setOpenWallet(selectedWallet);
openWalletImageItem = imageItem;
// display open wallet icon
try {

walletPanel.changeImage(imageItem,
                        new
URL(walletManager.getImageDir() +
selectedWallet.getOpenBitmap()));
} catch (Exception e) {
    System.out.println(e);
}

// get instruments
instrumentPanel.removeAllElements();
Enumeration instrumentNames =
    selectedWallet.getInstruments();
PaymentInstrument instrument;
String name;

while(instrumentNames.hasMoreElements()) {
    name =
(String)instrumentNames.nextElement();
    instrument =
selectedWallet.getInstrument(name);
    try {

instrumentPanel.addImage(name,
                        new
URL(walletManager.getImageDir() +
instrument.getBitmap()), instrument);
    } catch (Exception e) {
        System.out.println(e);
    }

if (payWindow.isPaymentRequest()) {
    // display instruments
    btnOK.setLabel("Select");
}
}

```



```

        ((CardLayout)walletDetail.getLayout()).
//          show(walletDetail,
"InstrumentSelection");
        displayChoosePaymentMethod();
    }
    else {
        payWindow.showMenu();
    }
}
else
    if (btnOK.getLabel() == "Select") {
        PaymentInstrument instrument;
        ImageItemRef imageItemRef =
instrumentPanel.getSelectedItems();
        instrument = (PaymentInstrument)
            imageItemRef.getUserObject();
        payWindow.authorizePayment(instrument);
    }
}
}
else
    if (evt.target == btnCancel) {
        if (payWindow.isPaymentRequest()) {
            payWindow.cancelTransaction();
        }
        else {
            payWindow.showMenu();
        }
    }
    return super.handleEvent(evt);
}
public void paint(Graphics g)
{
    g.drawLine(insets.left, size().height-insets.bottom,
        size().width-insets.right, size().height-
insets.bottom);
    g.drawLine(insets.left, size().height-insets.bottom+1,
        size().width-insets.right, size().height-
insets.bottom+1);
}
private void displayOpenWallet() {
    ImageItemRef imageItemRef = walletPanel.
        getSelectedItem();
    passwordEntryPanel.setWallet(imageItemRef.getName());
    btnOK.setLabel("Open");
    ((CardLayout)walletDetail.getLayout()).
        show(walletDetail, "PasswordEntryPanel");
    payWindow.setTitle("PayWindow - Open Wallet");
}
private void displayChoosePaymentMethod() {
    btnOK.setLabel("Select");
    btnOK.enable();
    ((CardLayout)walletDetail.getLayout()).
        show(walletDetail, "InstrumentSelection");
    payWindow.setTitle("PayWindow - Choose Payment Method");
}
}
}

```

```

-----
class PasswordEntryPanel extends Panel {
    private Image panelImage;
    private TextField password;
    // private URL imageFileURL;
    private Dimension panelSize;
    private Label enterPasswordLabel;
    private Label walletNameLabel;
    private WalletBrowserPanel walletBrowserPanel;
    public PasswordEntryPanel(WalletBrowserPanel wbp) {

        walletBrowserPanel = wbp;
        ImageLoader imageLoader = new ImageLoader(this);
        WalletManager wm = WalletManager.getWalletManager();
        try {
            // imageFileURL = new
            // URL("File:///c:/htdocs/PayWindow/JavaDemoDev/images/open.wallet.backgr
            // ound.gif");
            // URL("http://kimberly/PayWindow/JavaDemoDev/images/open.wallet.backgro
            // und.gif");

        } catch (Exception e) {
            System.out.println(e);
        }
        // panelImage = imageLoader.LoadImage(imageFileURL);
        panelImage = imageLoader.LoadImage(wm.getPwdScrImage());
        panelSize = new Dimension(panelImage.getWidth(this),
            panelImage.getHeight(this));
        enterPasswordLabel = new Label("Enter the Password for",
Label.CENTER);
        walletNameLabel = new Label("", Label.CENTER);
        password = new TextField();
        password.setEchoCharacter('*');
        setLayout(null);
        add(password);
        add(enterPasswordLabel);
        add(walletNameLabel);
    }
    public void setWallet(String walletName) {
        walletNameLabel.setText(walletName+":");
        password.setText("");
        walletBrowserPanel.disableOpenButton();
    }
    public void setPanelSize(Dimension size) {
        panelSize = size;
    }
    public String getPassword() {
        return password.getText();
    }
    public Dimension minimumSize() {
        return preferredSize();
    }
    public Dimension preferredSize() {
        return panelSize;
    }
}

```

```

public boolean handleEvent(Event evt) {
    if (evt.target == password) {
        if (evt.id == Event.KEY_RELEASE) {
            if (password.getText().length() == 0) {
                walletBrowserPanel.disableOpenButton();
            }
            else {
                walletBrowserPanel.enableOpenButton();
            }
        }
    }
    return super.handleEvent(evt);
}

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    // g.drawImage(panelImage, 0, 0, size().width,
    size().height, this);
    g.drawImage(panelImage,
        (size().width - panelImage.getWidth(this))/2,
        (size().height - panelImage.getHeight(this))/2,
        this);
    int labelWidth;
    FontMetrics fontMetrics = g.getFontMetrics();
    labelWidth = fontMetrics.stringWidth(
        enterPasswordLabel.getText()) + 60;
    enterPasswordLabel.reshape((size().width - labelWidth)/2,
        (size().height/2) - 30, labelWidth, 20);
    labelWidth = fontMetrics.stringWidth(
        walletNameLabel.getText()) + 60;
    walletNameLabel.reshape((size().width - labelWidth)/2,
        (size().height/2) - 10, labelWidth, 20);
    password.reshape((size().width - 100)/2,
        (size().height/2) + 10,
        100, 20);
    g.drawRect(0, 0, size().width-1, size().height-1);
}
}

```

### 認可 (a u t h o r i z a t i o n)

第11図は、好適具体例に従う認可表示スクリーンである。認可スクリーンは、取引の詳細を表示し、かつ一つの特別の取引の全ての詳細についてユーザが調べるのを容易にする。それはまた、ユーザが取引を受け入れ又はキャンセルし、或いは特別の取引のための支払方法を変更するのを可能にする。ウェブページ上の支払いボタンは、一つのウォレットが既に開かれているときこのスクリーンを始動する。支払い変更方法1100は、ユーザが異なる支払い装置を選択するのを可能にするために支払い装置スクリーンを表示し、かつ第9図に示された支

払いウインドー表示に制御を移す。オーダータブ1110は、オーダー、取引先又はアドレスに関する取引についての情報ページの表示を制御する。アクセプトボタン1140を押すと、ユーザの取引受け入れを信号し、かつそれは、取引データをウオーレットのレジスタに記録し、第12図に1200で示されるようにPAID受け入れを表示する。キャンセルボタン1150を押すと、第13図に1300で示されるように支払いキャンセルスクリーンを表示する。

#### 支払いウインドーメイン

第14図は、本発明の好適具体例に従い、メインスクリーンを例示している。このスクリーンは、ユーザがローカルモードでアプリケーションを起動しかつ支払いウインドーが管理モードで動作しているということを示すとき表示される。このアプリケーションは、デスクトップから起動する。スクリーンコントロールは、支払いウインドースクリーンを表示しかつユーザが新たなウオーレットを開き或いは作成するのを可能にするウオーレット選択ボタン1401を含んでいる。別のボタン、レポートボタン1410は、それが選択されるときレポートウインドーの一例を開く。レジスターボタン1420を選択すると、多様なレジスタ取引からユーザが選択するのを可能にするレジスターウインドーを開く。管理ボタン1430の選択は、ユーザが種々の管理タスクを実行するのを可能にする管理スクリーンを開く。もしユーザがバックアップ／回復ボタン1450を選択するならば、そのときバックアップ／回復ダイアログが開始される。最終的に、終了ボタン1460の選択は、スクリーンを閉じ、かつこのアプリケーションを終了させる。

#### 管理

第15図は、好適具体例に従う管理表示スクリーンの例示である。これは、ウオーレット、支払い方法及びユーザ選択設定に関する情報を入力するためのメインスクリーンである。この情報は、組織化され、そしてグループ化されたデータのページに提示される。管理表示は、ユーザによるボタンの選択に応答して第14図の管理ボタン1430から起動される。ウオーレットタブ1510は、現在アクティブなウオーレットに属する関連情報のページの管理を制御する。ユー

ザは、情報ページを変更するためにマウス又はキーボードを利用することができる。完了ボタン1590は、それが選択されるときウオーレット内に新たな情報を保存する。キャンセルボタン1595は、それが選択されるとき、この変更を捨て去り、表示スクリーンを閉じる。

第16図は、好適具体例に従う管理ウオーレットページの表示を例示している。ユーザは、ウオーレットの名前をエントリーフィールド1660に入力しなければならない、ウオーレットの所有者は1670に入力されなければならない、名前を付けたウオーレットに相当するパスワードは、1675に入力されなければならない、パスワードエントリーは、パスワードの英字及び数字の代わりに入力された文字と共に表示され、かつ確認パスワードがエントリフィールド1678に入力されなければならない。確認パスワード1678は、1675で入力されたパスワードに一致しなければならない。ブラウズボタン1680は、ファイルオープンダイアログを開いて、ウオーレットのリストをユーザに提示して、このプロセスへの入力として使用するウオーレットアイコンファイルから選択する。ウオーレット削除ボタン1685は、現在のウオーレットをこのシステムから取り除き、かつ削除の前に確認メッセージボックスを表示する。

第15図の支払い管理タブ1520は、第17図に示されるようにスクリーンを開始する。第17図は、好適具体例に従う支払い管理ウインドーである。ユーザは、支払い方法の情報を観察し又は編集するために管理スクリーンの支払いページを使用することができる。

リストボックスコントロール1710は、このウオーレット内に現在インストールされている全ての支払い方法を表示する。ユーザは、リストボックスコント

ロールをスクロールして、所有者に利用可能の全ての支払い方法1760を見ることができる。ユーザは、その情報を見る前に支払い方法を選択しなければならない。この選択は、ユーザが必要とする支払い方法上にカーソルを位置決めし、かつマウスボタンをクリックするか又は支払方法としてそのカードを選択するための入力キーを押すことによって達成される。カードネーム編集フィールド1750は、このシステムが支払いのために現在選択した方法を参照するために利用

する名前を表示し又は編集するために使用される。カードタイプデータエントリフィールド1760は、支払方法の種々のタイプに相当する。このタイプの多くは、マスターカード、VISA、ディスカバーのように、ユーザに対してあらかじめ定められている。

関連したカード番号エントリフィールド1780はまた、カード番号を表示しかつ／又は編集するために備えられている。エントリフィールド1782はまた、カードの終了日を表示しかつ／又は編集するために備えられている。証明書ダイアログボックス1784ボタンがまた、付属書の開始、即ち証明書の編集のために備えられている。チェックボックスコントロール1792がまた、もしユーザが選択したカードに特別のウオーレットのためのデフォルトの支払方法を望むならば、備えられる。ウオーレットが開かれる毎に特別の支払方法を選択する必要があるないので、これは魅力的な特徴である。支払い方法のホルダー（即ち、ウオーレット、財布、ブリーフケース、スマートカード）が選択されるとき、お気に入りのクレジットカードが通常容易な読み出しのために便利なロケーションに記憶されているユーザのウオーレットを並べるために、人間工学が十分考慮されている。2つの他のボタンが、支払方法のホルダーに支払方法を付加し1786、かつそこから削除するために1790備えられている。削除ボタン1790は、支払い方法の削除の前に確認メッセージを表示する。プロセスの完了1742を意味するボタン及びプロセスをキャンセル1746するボタンがまた供えられる。

第18図は、好適具体例に従う管理選択ページの例示である。アドレスライン1エディットコントロール1810が、特別のユーザのために第一のシップメントアドレスラインを観察し、又は編集するために使用される。アドレスライン2

エディットコントロール1820が、特別のユーザのために第二のシップメントアドレスラインを観察し又は編集するために使用される。シティクレジットカードコントロール1830が、特別のユーザのためにシップメントシティを編集し又は観察するために使用される。ステートエディットコントロール1850は、特別のユーザのためのシップメントステートを編集し又は観察するために使用される